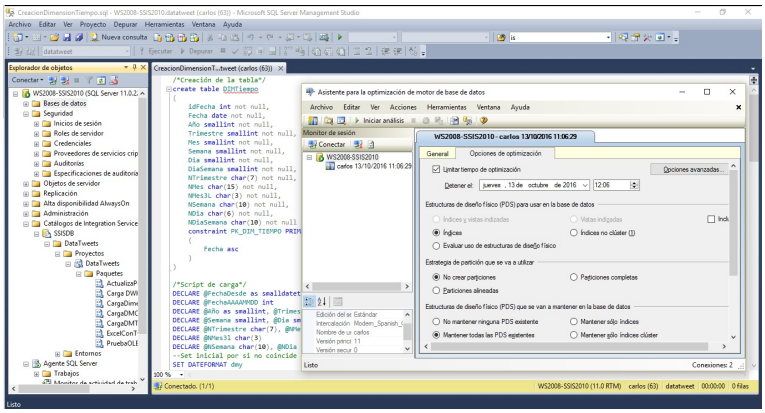
APPUNTI INFORMATICA (PIETRO VACCARI)

Conoscere ed utilizzare SQL Server

SQL Server è un database relazionale di Microsoft, che consente di archiviare, gestire e analizzare grandi quantità di dati. Di seguito spiego alcuni concetti chiave relativi a SQL Server e alla sua gestione.



1. Confronto tra le versioni di SQL Server:
   SQL Server è disponibile in diverse versioni, ognuna delle quali ha funzionalità e limitazioni diverse. Le principali versioni di SQL Server sono:
   a. SQL Server Express: versione gratuita con funzionalità di base e limitazioni sui dati archiviati e sulle prestazioni.
   b. SQL Server Standard: versione a pagamento con funzionalità avanzate, che consente di gestire grandi quantità di dati.

c. SQL Server Enterprise: versione a pagamento con funzionalità avanzate per la gestione di grandi quantità di dati e per l'elaborazione analitica.

d. SQL Server Developer: versione gratuita per gli sviluppatori, con funzionalità simili a quelle della versione Enterprise.

e. SQL Server Web: versione a pagamento destinata alla pubblicazione di applicazioni web.

2. Installazione e configurazione:
   L'installazione di SQL Server può essere effettuata tramite il programma di installazione di Microsoft. Durante l'installazione, è possibile scegliere le funzionalità da installare, configurare le opzioni di sicurezza e configurare i servizi di rete.
3. SQL Server Management Studio:
   SQL Server Management Studio è un'applicazione che consente di gestire e amministrare un'istanza di SQL Server. È possibile creare database, tabelle, viste, funzioni, trigger e stored procedure, nonché eseguire query e monitorare le prestazioni del server.
4. Configurazione dei DB:
   È possibile configurare i database in SQL Server tramite SQL Server Management Studio. È possibile definire le tabelle, le viste, gli indici e le relazioni tra le tabelle.
5. Creazione di tabelle, viste e indici:
   La creazione di tabelle, viste e indici è possibile tramite SQL Server Management Studio. È possibile definire i campi delle tabelle, i tipi di dati, le relazioni tra le tabelle e gli indici.
6. Funzioni, trigger e stored procedure:
   Le funzioni, i trigger e le stored procedure sono oggetti del database che consentono di automatizzare le operazioni di gestione dei dati. Le stored procedure sono eseguite sul server e possono accedere ai dati archiviati nel database.
7. Gestione della sicurezza:
   La gestione della sicurezza in SQL Server può essere effettuata tramite SQL Server Management Studio. È possibile definire gli utenti e i gruppi di utenti, assegnare i diritti di accesso e definire le regole di sicurezza.
8. Backup e recupero:
   SQL Server consente di eseguire il backup e il ripristino dei dati archiviati nel database. È possibile effettuare il backup dei dati tramite SQL Server Management Studio o utilizzando strumenti di backup di terze parti.
9. Porting dei dati:
   Il porting dei dati consente di migrare i dati da un database a un altro. È possibile utilizzare strumenti di porting di terze parti o utilizzare le funzionalità di importazione ed esportazione di SQL Server.
10. Gestione ed amministrazione di sistema:
    La gestione ed amministrazione di sistema in SQL Server è una parte molto importante per garantire la continuità delle attività e la sicurezza dei dati. In questo contesto, il lavoro dell'amministratore di sistema comprende le seguenti attività:
    - Installazione di SQL Server: l'installazione può essere fatta in diverse modalità, come ad esempio l'installazione in modalità "basic" o l'installazione avanzata con molte opzioni personalizzabili.
    - Configurazione di SQL Server: dopo l'installazione, il server può essere configurato per funzionare in modo specifico per le esigenze dell'applicazione e dell'ambiente in cui viene utilizzato.
    - Monitoraggio delle prestazioni: l'amministratore deve essere in grado di monitorare le prestazioni del server per individuare eventuali problemi e garantire una risposta rapida in caso di problemi.
    - Ottimizzazione delle prestazioni: l'amministratore deve avere la capacità di identificare e risolvere i problemi di prestazioni.
    - Gestione della capacità: l'amministratore deve essere in grado di gestire la capacità del server, pianificando eventuali espansioni o aggiornamenti.
    - Gestione delle patch: l'amministratore deve tenere il server sempre aggiornato con le ultime patch di sicurezza e funzionalità.
    - Backup e ripristino: l'amministratore deve pianificare e gestire le attività di backup e ripristino dei dati, in modo da garantire la continuità del servizio anche in caso di problemi.

11. Pianificazione delle operazioni:
    La pianificazione delle operazioni è un'attività fondamentale per garantire la continuità del servizio e la disponibilità dei dati. Tra le attività principali di pianificazione delle operazioni troviamo:
    - Pianificazione dei backup: l'amministratore deve pianificare le attività di backup dei dati in modo da garantire la disponibilità dei dati anche in caso di problemi.
    - Pianificazione delle attività di manutenzione: l'amministratore deve pianificare le attività di manutenzione del server, come ad esempio la deframmentazione dei dischi.
    - Pianificazione degli aggiornamenti: l'amministratore deve pianificare gli aggiornamenti del server, garantendo la continuità del servizio anche durante l'aggiornamento.
12. Cenni su transazioni:
    Le transazioni sono utilizzate per garantire l'integrità dei dati e per garantire l'atomicità delle operazioni. Una transazione rappresenta una serie di operazioni che devono essere eseguite tutte o nessuna. In caso di problemi durante l'esecuzione della transazione, questa viene automaticamente annullata e il sistema viene riportato allo stato precedente.
13. Cenni a problemi di load balancing:
    Il load balancing è l'attività di distribuire il carico di lavoro su più server, in modo da garantire la disponibilità del servizio anche in caso di picchi di carico. SQL Server supporta il load balancing attraverso la funzionalità di clustering.
14. In SQL Server, la sicurezza può essere gestita a livello di istanza e di database. A livello di istanza, è possibile configurare le impostazioni di sicurezza per l'intero server SQL, ad esempio tramite l'Autenticazione di Windows o l'Autenticazione di SQL Server.

    A livello di database, la sicurezza può essere gestita tramite utenti e ruoli. Gli utenti possono essere creati per accedere a un database specifico e assegnati a ruoli specifici che definiscono i permessi di accesso alle risorse del database. È possibile concedere a un utente l'accesso solo alle risorse necessarie per svolgere il proprio lavoro, limitando così il rischio di accesso non autorizzato alle informazioni.
15. Il monitoraggio delle risorse è un aspetto importante dell'amministrazione del database in SQL Server. Questo include il monitoraggio delle prestazioni del server, dei database e dei processi in esecuzione. È possibile utilizzare strumenti come SQL Server Profiler o Dynamic Management Views (DMV) per monitorare le query in esecuzione e identificare eventuali problemi di prestazioni.

    Inoltre, SQL Server offre funzionalità di load balancing per garantire la scalabilità dei database. È possibile distribuire i database su diversi server e utilizzare la replica per garantire la disponibilità dei dati. In questo modo, è possibile distribuire il carico di lavoro su più server per evitare il sovraccarico del server e migliorare le prestazioni complessive del sistema.

Database e SQL

Linguaggio SQL:

SQL (Structured Query Language) è un linguaggio di programmazione utilizzato per gestire i database relazionali. SQL consente agli utenti di creare, modificare e manipolare i dati all'interno di un database.

Gli elementi principali di SQL sono:

- Data Definition Language (DDL): utilizzato per creare, modificare e eliminare gli oggetti del database, come tabelle, indici, viste, procedure e trigger.
- Data Manipulation Language (DML): utilizzato per aggiungere, modificare e eliminare i dati all'interno delle tabelle.
    - Query Language: utilizzato per interrogare il database e recuperare i dati che soddisfano determinati criteri.
    - Transaction Control Language (TCL): utilizzato per gestire le transazioni, ovvero un insieme di operazioni eseguite su un database come un'unità atomica.

In SQL, una tabella è composta da una serie di righe e colonne che contengono i dati. Una riga rappresenta un record nel database, mentre una colonna rappresenta un attributo del record.

Le query SQL sono utilizzate per recuperare i dati dal database. La query SELECT è la più comune ed è utilizzata per selezionare una o più colonne da una o più tabelle. Le query SELECT possono essere filtrate utilizzando la clausola WHERE per selezionare solo i record che soddisfano determinati criteri.

Ad esempio, la seguente query seleziona tutti i record dalla tabella "Utenti" dove il campo "Nome" è uguale a "Mario":

```
SELECT * FROM Utenti WHERE Nome = 'Mario';
```

DDL

DDL (Data Definition Language) è un insieme di comandi SQL che consentono di definire la struttura e la composizione dei dati all'interno di un database. I principali comandi DDL sono:

1. CREATE TABLE: consente di creare una nuova tabella all'interno del database, specificando il nome della tabella, i nomi e i tipi di dati delle colonne e le eventuali chiavi primarie e straniere.

   Esempio:

   ```
   CREATE TABLE prodotti (
       id_prodotto INT(11) NOT NULL AUTO_INCREMENT,
       nome VARCHAR(50) NOT NULL,
       descrizione TEXT,
       prezzo DECIMAL(10,2) NOT NULL,
       id_categoria INT(11) NOT NULL,
       PRIMARY KEY (id_prodotto),
       FOREIGN KEY (id_categoria) REFERENCES categorie(id_categoria)
   );
   ```

   Questo comando crea una nuova tabella chiamata "prodotti", con una chiave primaria "id_prodotto" di tipo INT, un campo "nome" di tipo VARCHAR, un campo "descrizione" di tipo TEXT, un campo "prezzo" di tipo DECIMAL e una chiave esterna "id_categoria" che fa riferimento alla tabella "categorie".

2. ALTER TABLE: consente di modificare la struttura di una tabella già esistente, aggiungendo o rimuovendo colonne, definendo o eliminando vincoli o modificando il tipo di dati delle colonne.

   Esempio:

   ```
   ALTER TABLE prodotti
   ADD COLUMN quantita INT(11) NOT NULL,
   ADD CONSTRAINT CHK_quantita CHECK (quantita >= 0);
   ```

   Questo comando aggiunge una nuova colonna chiamata "quantità" di tipo INT alla tabella "prodotti" e definisce un vincolo di controllo ("CHECK") che impone il valore minimo di 0 per la quantità.

3. DROP TABLE: consente di eliminare una tabella dal database.

   Esempio:

   ```
   DROP TABLE prodotti;
   ```

   Questo comando elimina la tabella "prodotti" dal database.

DML

DML (Data Manipulation Language) è un sottoinsieme del linguaggio SQL utilizzato per manipolare i dati all'interno di un database. I comandi DML includono:

- SELECT: utilizzato per recuperare i dati dal database
- INSERT: utilizzato per inserire nuovi dati in una tabella del database
- UPDATE: utilizzato per modificare i dati già presenti in una tabella del database

- DELETE: utilizzato per eliminare i dati da una tabella del database

Ecco alcuni esempi di come utilizzare i comandi DML in SQL:

```
SELECT:
SELECT * FROM tabella;
SELECT colonna1, colonna2 FROM tabella WHERE condizione;
```

```
INSERT:
INSERT INTO tabella (colonna1, colonna2) VALUES (valore1, valore2);
```

```
UPDATE:
UPDATE tabella SET colonna1 = valore1, colonna2 = valore2 WHERE condizione;
```

```
DELETE:
DELETE FROM tabella WHERE condizione;
```

Il Join viene utilizzato per combinare i dati da due o più tabelle in base ad una relazione tra di esse. Esistono diversi tipi di Join, come l'Inner Join, il Right Join e il Left Join, ciascuno dei quali restituisce un risultato diverso a seconda delle relazioni tra le tabelle.

L'ordinamento dei dati può essere effettuato tramite la clausola ORDER BY, che può essere utilizzata per ordinare i dati in base ad uno o più campi, sia in modo ascendente che discendente.

Gli operatori Aggregati (Count, Sum, Max, Min, Avg) vengono utilizzati per eseguire operazioni matematiche sui dati selezionati, restituendo il valore aggregato. Questi operatori sono spesso utilizzati in combinazione con la clausola GROUP BY, che viene utilizzata per raggruppare i dati in base ad uno o più campi.

La clausola HAVING viene utilizzata per filtrare i risultati di una query in base ad una condizione specificata, ed è spesso utilizzata in combinazione con la clausola GROUP BY.

Infine, le subQuery sono una tecnica avanzata che permette di eseguire una query all'interno di un'altra query, permettendo di effettuare interrogazioni complesse e di selezionare solo i dati che soddisfano determinate condizioni.

Le query SQL sono costituite da diverse parti fondamentali, tra cui:

- SELECT: utilizzata per selezionare i campi che si desidera visualizzare nell'output della query.
- FROM: utilizzata per specificare la o le tabelle da cui recuperare i dati.
- WHERE: utilizzata per filtrare i dati in base a determinati criteri.
- GROUP BY: utilizzata per raggruppare i dati in base a determinati attributi.
- ORDER BY: utilizzata per ordinare i dati in base a determinati attributi.
- JOIN: è un'operazione che consente di combinare le righe di due o più tabelle in base a una colonna comune. Ci sono diversi tipi di join disponibili, tra cui Inner Join, Right Join e Left Join.
    - Inner Join: l'Inner Join restituisce solo le righe che corrispondono alle condizioni specificate nella clausola di join. In altre parole, restituisce solo le righe che hanno una corrispondenza nelle due tabelle coinvolte nella join.
    - Right Join: il Right Join restituisce tutte le righe della seconda tabella (la tabella a destra nella clausola di join) e solo le righe della prima tabella (la tabella a sinistra nella clausola di join) che soddisfano le condizioni specificate. Se una riga della seconda tabella non ha una corrispondenza nella prima tabella, il risultato della join conterrà NULL nella colonna corrispondente.
    - Left Join: il Left Join restituisce tutte le righe della prima tabella (la tabella a sinistra nella clausola di join) e solo le righe della seconda tabella (la tabella a destra nella clausola di join) che soddisfano le condizioni specificate. Se una riga della prima tabella non ha una corrispondenza nella seconda tabella, il risultato della join conterrà NULL nella colonna corrispondente.

ESEMPI:

1. Selezionare tutti i dati dalla tabella "clienti":

```
SELECT * FROM clienti;
```

2. Selezionare solo i nomi dei clienti dalla tabella "clienti" ordinati in ordine alfabetico crescente:

```
SELECT nome FROM clienti ORDER BY nome ASC;
```

3. Selezionare solo i prodotti con un prezzo maggiore di 50€ dalla tabella "prodotti":

```
SELECT * FROM prodotti WHERE prezzo > 50;
```

4. Selezionare solo i prodotti con un prezzo maggiore di 50€ e con uno stock maggiore di 10 dalla tabella "prodotti":

```
SELECT * FROM prodotti WHERE prezzo > 50 AND stock > 10;
```

5. Selezionare il numero totale di prodotti venduti e il loro prezzo medio dalla tabella "vendite":

```
SELECT COUNT(*), AVG(prezzo) FROM vendite;
```

6. Selezionare il numero totale di vendite effettuate da ogni cliente dalla tabella "vendite":

```
SELECT cliente, COUNT(*) FROM vendite GROUP BY cliente;
```

QL

Il Query Language (QL) in SQL è il linguaggio utilizzato per interrogare e manipolare i dati all'interno di un database relazionale. Consente di recuperare informazioni specifiche da una o più tabelle, selezionando determinati campi, filtrando i dati in base a specifici criteri e raggruppando le informazioni in base a determinati attributi.

TCL

Transaction Control Language (TCL) è utilizzato per controllare il modo in cui vengono eseguite le transazioni e per garantire che i dati del database rimangano coerenti e accurati. I comandi principali di TCL sono COMMIT, ROLLBACK e SAVEPOINT. COMMIT viene utilizzato per confermare una transazione, mentre ROLLBACK viene utilizzato per annullare una transazione. SAVEPOINT viene utilizzato per creare un punto di ripristino all'interno di una transazione, in modo che possa essere annullata fino a quel punto.

Normalizzazione (anomalie, forme normali e dipendenze funzionali in un database)

Questo codice è un esempio di script PHP che gestisce l'input dei dati di un form HTML e li inserisce in un database MySQL.

In particolare, il codice accede ai dati inseriti dall'utente nel form HTML attraverso l'uso di variabili PHP e il metodo POST. Quindi, dichiara le variabili per il nome del server, l'utente del database, la password e il nome del database stesso, per poi stabilire una connessione al database utilizzando la classe mysqli di PHP.

Successivamente, viene costruita una query SQL per inserire i dati forniti dall'utente nella tabella del database specificata. La query è costruita concatenando i valori delle variabili utilizzando l'operatore di stringa di concatenazione ".".

Dopo che la query è stata eseguita, il codice controlla se l'operazione è stata eseguita con successo o se ci sono stati errori. Se l'operazione ha avuto successo, viene visualizzato un messaggio di conferma, altrimenti viene visualizzato un messaggio di errore insieme alla descrizione dell'errore.

Infine, il codice visualizza un riepilogo dei dati che l'utente ha inserito nel form HTML, utilizzando le variabili dichiarate all'inizio dello script.

La normalizzazione è un processo di progettazione dei database relazionali volto a minimizzare le anomalie e a garantire l'integrità dei dati. Ci sono diverse forme normali che descrivono la corretta organizzazione dei dati in una tabella, in base alle dipendenze funzionali tra gli attributi.

La prima forma normale (1NF) richiede che ogni attributo di una tabella abbia un valore atomico e non ripetuto, ovvero che non ci siano attributi multivalore. Inoltre, deve essere presente una chiave primaria unica per ogni riga della tabella.

La seconda forma normale (2NF) richiede che ogni attributo non chiave sia funzionalmente dipendente dall'intera chiave primaria, e non solo da una parte di essa. In altre parole, ogni attributo deve dipendere interamente dalla chiave primaria, e non da una parte di essa.

La terza forma normale (3NF) richiede che ogni attributo non chiave sia dipendente solo dalla chiave primaria, e non da altri attributi non chiave. In altre parole, non ci devono essere dipendenze funzionali tra gli attributi non chiave.

Esistono anche forme normali più avanzate, come la forma normale di Boyce-Codd (BCNF) e la quarta forma normale (4NF), che sono utilizzate per garantire un'ulteriore integrità dei dati e una maggiore efficienza nell'accesso ai dati.

MySQL

MySQL è un sistema di gestione di database relazionale open source che utilizza il linguaggio SQL (Structured Query Language) per la manipolazione dei dati.

Per installare MySQL, è necessario scaricare il pacchetto di installazione dal sito ufficiale di MySQL, quindi seguire le istruzioni per l'installazione. È anche possibile utilizzare strumenti come XAMPP o WAMP, che consentono di installare MySQL e altri strumenti necessari come Apache e PHP in un'unica installazione.

Dopo aver installato MySQL, è possibile creare tabelle utilizzando il comando CREATE TABLE. Ad esempio, per creare una tabella "clienti" con tre colonne "ID", "nome" e "cognome", la query sarebbe:

```
CREATE TABLE clienti (
    ID INT NOT NULL,
    nome VARCHAR(50) NOT NULL,
    cognome VARCHAR(50) NOT NULL,
    PRIMARY KEY (ID)
);
```

Per effettuare una query in MySQL, è possibile utilizzare il comando SELECT. Ad esempio, per selezionare tutti i record dalla tabella "clienti", la query sarebbe:

```
SELECT * FROM clienti;
```

Per amministrare MySQL, è possibile utilizzare strumenti come MySQL Workbench o la riga di comando di MySQL. Questi strumenti consentono di gestire gli utenti, i permessi, le tabelle e i dati. È anche possibile utilizzare la replica di MySQL per creare copie di backup dei dati e migliorare la disponibilità.

La sicurezza in MySQL può essere gestita tramite l'utilizzo di utenti e privilegi. È possibile creare utenti e assegnare loro diversi livelli di accesso ai database e alle tabelle. Inoltre, MySQL supporta la crittografia dei dati in transito e a riposo per garantire la sicurezza dei dati.

In confronto a SQL Server, MySQL è un sistema di gestione di database relazionale open source che può essere utilizzato gratuitamente. Tuttavia, SQL Server offre una maggiore scalabilità e prestazioni, nonché strumenti di amministrazione più avanzati come SQL Server Management Studio.

Per inserire una foreign key su MySQL, puoi utilizzare il comando ALTER TABLE con la clausola ADD FOREIGN KEY, seguita dal nome della colonna che devi associare alla chiave primaria nella tabella di riferimento. Ad esempio, supponiamo di avere due tabelle "Ordine" e "Cliente", dove "Ordine" ha una colonna "cliente_id" che deve essere una foreign key che si riferisce alla colonna "id" nella tabella "Cliente". Il codice per creare la foreign key sarebbe il seguente:

```
ALTER TABLE Ordine
ADD CONSTRAINT N_vincolo
FOREIGN KEY (cliente_id) REFERENCES Cliente(id);
```

Questo comando indica a MySQL di aggiungere una foreign key alla colonna "cliente_id" nella tabella "Ordine", che fa riferimento alla colonna "id" nella tabella "Cliente". Se la tabella di riferimento non esiste o non ha una chiave primaria corrispondente, verrà generato un errore. Inoltre, se ci sono già valori nella colonna "cliente_id" che non corrispondono a nessuna riga nella tabella "Cliente", l'aggiunta della foreign key potrebbe fallire a causa di violazioni di integrità referenziale.

Ecco alcuni dei principali comandi MySQL che puoi utilizzare per gestire un database:

1. CREATE DATABASE: consente di creare un nuovo database.
2. DROP DATABASE: consente di eliminare un database esistente.
3. USE DATABASE: consente di selezionare un database per l'utilizzo.
4. CREATE TABLE: consente di creare una nuova tabella all'interno del database selezionato.
5. DROP TABLE: consente di eliminare una tabella esistente.
6. ALTER TABLE: consente di modificare la struttura di una tabella esistente, come l'aggiunta o la rimozione di colonne o la modifica di proprietà di colonna.
7. INSERT INTO: consente di inserire nuovi dati in una tabella.
8. SELECT: consente di recuperare dati dalla tabella.
9. UPDATE: consente di aggiornare i dati esistenti nella tabella.
10. DELETE: consente di eliminare i dati dalla tabella.
11. JOIN: consente di combinare dati da più tabelle in base a una relazione tra le tabelle.
12. INDEX: consente di creare un indice su una o più colonne della tabella, per migliorare le prestazioni delle query.

Per scrivere query complesse su MySQL, è necessario avere una conoscenza approfondita del linguaggio SQL e delle funzionalità di MySQL. Ecco alcuni consigli utili per scrivere query complesse su MySQL:

1. Pianificare la query: prima di scrivere la query, è importante capire esattamente cosa si vuole ottenere dai dati. Identificare le tabelle coinvolte, le colonne interessate e le condizioni di filtraggio.
2. Utilizzare JOIN: per recuperare dati da più tabelle, è necessario utilizzare JOIN. MySQL supporta diverse tipologie di JOIN, come INNER JOIN, LEFT JOIN, RIGHT JOIN e FULL OUTER JOIN. È importante scegliere il tipo di JOIN corretto in base alle esigenze della query.
3. Utilizzare le funzioni di aggregazione: le funzioni di aggregazione, come SUM, COUNT, AVG e MAX, consentono di eseguire calcoli sui dati della tabella. È importante utilizzare le funzioni di aggregazione corrette per ottenere i risultati desiderati.
4. Utilizzare le sottoquery: le sottoquery consentono di eseguire una query all'interno di un'altra query. Sono utili per eseguire operazioni di filtro e raggruppamento avanzate sui dati.
5. Utilizzare gli indici: gli indici possono migliorare le prestazioni della query. È importante creare gli indici corretti sulle colonne utilizzate nelle clausole WHERE e JOIN per migliorare le prestazioni della query.
6. Utilizzare le istruzioni di controllo di flusso: le istruzioni di controllo di flusso, come IF e CASE, consentono di eseguire operazioni condizionali sui dati della tabella. Sono utili per eseguire operazioni complesse sui dati.

Ecco alcuni esempi di codice per scrivere query complesse su MySQL:

1. JOIN:

```
SELECT orders.order_id, customers.customer_name
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id;
```

Questa query recupera l'ID dell'ordine e il nome del cliente associato all'ordine da due tabelle diverse, "orders" e "customers", utilizzando INNER JOIN per combinare i dati.

2. Funzioni di aggregazione:

```
SELECT SUM(order_amount) as total_amount, COUNT(*) as total_orders
FROM orders;
```

Questa query calcola la somma totale dell'importo dell'ordine e il numero totale di ordini nella tabella "orders" utilizzando le funzioni di aggregazione SUM e COUNT.

3. Sottoquery:

```
SELECT customer_name
FROM customers
WHERE customer_id IN (
   SELECT customer_id
   FROM orders
   WHERE order_date BETWEEN '2022-01-01' AND '2022-12-31'
);
```

Questa query recupera i nomi dei clienti che hanno effettuato un ordine tra il 1 gennaio 2022 e il 31 dicembre 2022, utilizzando una sottoquery per recuperare l'elenco dei clienti che hanno effettuato un ordine in quel periodo.

4. Indici:

```
SELECT *
FROM orders
WHERE customer_id = 1234;
```

Questa query recupera tutti gli ordini associati a un cliente specifico utilizzando l'indice sulla colonna "customer_id" per migliorare le prestazioni della query.

5. Istruzioni di controllo di flusso:

```
SELECT customer_name, IF(order_amount > 1000, 'High Value', 'Low Value') as order_value
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id;
```

Questa query recupera il nome del cliente e il valore dell'ordine (alto o basso) associato a ciascun ordine, utilizzando l'istruzione IF per eseguire operazioni condizionali sulla colonna "order_amount".

PHP

In PHP, è possibile interagire con un database MySQL utilizzando l'estensione MySQLi (MySQL improved), che offre una serie di funzioni per connettersi, eseguire query e gestire i risultati, ad una pagina web.

Per creare una pagina PHP per interagire con un database MySQL, si può seguire il seguente processo:

1. Connessione al database: per connettersi a un database MySQL si può utilizzare la funzione `mysqli_connect()`, che richiede di specificare l'host, il nome utente e la password per 'accesso al database. Ad esempio:

```
<?php
$host = "localhost";
$username = "user";
$password = "password";
$dbname = "database";
```

```
$conn = mysqli_connect($host, $username, $password, $dbname);

// Verifica la connessione
if (!$conn) {
    die("Connessione al database fallita: " . mysqli_connect_error());
}
echo "Connessione al database stabilita con successo";
?>
```

2. Esecuzione di una query: una volta stabilita la connessione, si può eseguire una query SQL utilizzando la funzione `mysqli_query()`. Ad esempio:

```
$sql = "SELECT * FROM users";
$result = mysqli_query($conn, $sql);

// Verifica se la query ha restituito dei risultati
if (mysqli_num_rows($result) > 0) {
    // Itera sui risultati e li stampa a video
    while($row = mysqli_fetch_assoc($result)) {
        echo "ID: " . $row["id"]. " - Nome: " . $row["name"]. " - Email: " . $row["email"]. "<br>";
    }
} else {
    echo "Nessun risultato trovato";
}
```

3. Chiusura della connessione: una volta completate le operazioni, è importante chiudere la connessione al database utilizzando la funzione `mysqli_close()`. Ad esempio:

```
mysqli_close($conn);
```

Per passare i valori tra pagine, si possono utilizzare le variabili di sessione o le query string dell'URL. Ad esempio, per passare un parametro `id` alla pagina successiva:

```
<a href="pagina2.php?id=123">Vai a pagina 2</a>
```

Nella pagina successiva, il parametro può essere recuperato utilizzando la variabile superglobale `$_GET`:

```
$id = $_GET["id"];
```

Infine, per visualizzare i risultati delle query su una pagina PHP, si possono utilizzare le funzioni di output come `echo` o `print`, oppure è possibile utilizzare il markup HTML per creare tabelle o altri elementi grafici.

Eempio PHP per l’aggiunta di dati in un db da una pagina HTML

HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>INSERIMENTO DATI</title>
    </head>
    <style>
        .text
        {
            text-align: center;
        }
        .bottomright
        {
            position: absolute;
            bottom: 8px;
            right: 16px;
            font-size: 18px;
        }
```

```
    </style>
    <body bgcolor="gray">
        <table align="center" border="2" bordercolor="black" cellspacing="0" cellpadding="4">
            <tr>
                <td><b><h3>&nbspREGISTRAZIONE DATI: </h3></b>
                </td>
            </tr>
        </table>
        <br><br>
        <div align="center">
        <form action="aggiungi2.php" method="post">
        <table border="1" bordercolor="black" class="text" cellspacing="0" cellpadding="5">
        <tr>
            <td>
                <b class="text">Nome:</b>
            </td>
            <td>
                <input type="text" name="nome" size="20" maxlength="40" value="">
            </td>
        </tr>
        <tr>
            <td>
                <b>Cognome:</b>
            </td>
            <td>
                <input type="text" name="cognome" size="20" maxlength="40" value="">
            </td>
        </tr>
        <tr>
            <td>
                <b>Email:</b>
            </td>
            <td>
            <input type="text" name="email" size="20" maxlength="40" value="">
            </td>
        </tr>
        <tr>
            <td>
                <b>Codice fiscale:</b>
            </td>
            <td>
                <input type="text" name="codice_fiscale" size="20" maxlength="40" value="">
            </td>
        </tr>
        <tr>
            <td>
                <b>Reg_date:</b>
            </td>
            <td>
                <input type="date" name="reg_date" size="20" maxlength="40" value="">
            </td>
        </tr>
    <table border="1" bordercolor="black" cellspacing="0" cellpadding="5">
    <br><br><br>
        <tr>
            <td>
                <input type="submit" value="Invia i Dati" name="BInvia">
            </td>
            <td>
                <input type="reset" value="Annulla" name="BReset">
            </td>
        </tr>
    </table>
    <p class="bottomright">Esercizio di Pietro Vaccari</p>
    </form>
    </body>
</html>
```

Questo codice HTML mostra una pagina web che ha un form con campi di input per il nome, cognome, email, codice fiscale e data di registrazione di un utente. Il form ha un pulsante di invio e un pulsante di reset per cancellare i dati inseriti.

Il form invia i dati a un file PHP chiamato "aggiungi2.php" attraverso il metodo POST. Quando l'utente fa clic sul pulsante di invio, i dati vengono inviati al file PHP per essere elaborati.

La pagina ha anche alcune proprietà di stile CSS, come l'allineamento del testo al centro, la posizione in basso a destra di un testo e il colore di sfondo grigio. Inoltre, il file include un'intestazione HTML con il titolo "INSERIMENTO DATI".

PHP

```
<?php
    echo"<body bgcolor='lightgreen'>";$nome=$_POST['nome'];
    $nome = $_POST['nome'];
    $cognome = $_POST['cognome'];
    $email = $_POST['email'];
    $codice_fiscale = $_POST['codice_fiscale'];
    $reg_date = $_POST['reg_date'];

    $servername="localhost";
    $username="root";
    $password="";
    $dbname="esercizio_php";
    $conn = new mysqli($servername, $username, $password, $dbname);
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }
    $sql = "INSERT INTO Tabella_2 (nome , cognome, email , codice_fiscale , reg_date) values ('$nome' , '$cognome' , '$email' , '$codice_fi

    if ($conn->query($sql) === TRUE) {
        echo "AGGIORNAMENTO RIUSCITO";
    } else {
        echo "Error " . $conn->error;
    }
    echo"<br><br><br>Ecco il riepilogo dei dati che ha inserito:<br>
    Cognome e nome: $cognome $nome<br>
    Email: $email<br>
    Codice Fiscale: $codice_fiscale<br>
    Data registazione: $reg_date<br>";
    $conn->close();
    ?>
```

Wordpress

Wordpress è un sistema di gestione dei contenuti (CMS) open-source molto popolare che consente agli utenti di creare e gestire siti web. Di seguito sono riportati alcuni concetti fondamentali di Wordpress:

- Introduzione a Wordpress: Wordpress è un CMS che consente agli utenti di creare e gestire siti web. Wordpress ha un'interfaccia utente intuitiva e una vasta gamma di temi e plugin per personalizzare il design e le funzionalità del sito web.
- Installazione e configurazione: Wordpress è relativamente facile da installare e configurare. Ci sono due modi principali per installare Wordpress: manualmente o tramite un provider di hosting che offre un'installazione automatica di Wordpress. Una volta installato, Wordpress richiede una configurazione iniziale, ad esempio l'impostazione del nome del sito web, dell'URL e dell'utente amministratore.
- Componenti principali e loro utilizzo: I componenti principali di Wordpress includono i temi, i plugin e i widget. I temi determinano l'aspetto del sito web, mentre i plugin forniscono funzionalità aggiuntive. I widget sono elementi che possono essere posizionati nelle sidebar o in altre aree del sito web per visualizzare informazioni specifiche. Wordpress ha anche un'area amministrativa (detta "Dashboard") che consente agli utenti di gestire il contenuto del sito web, ad esempio la creazione di post e pagine, la modifica del design e la gestione dei commenti degli utenti.

In sintesi, Wordpress è un CMS popolare e facile da usare che consente agli utenti di creare e gestire siti web con una vasta gamma di funzionalità e personalizzazioni disponibili tramite temi e plugin.

Sviluppo di un sito Wordpress

Lo sviluppo individuale di un sito su WordPress può essere suddiviso in diverse fasi:

1. Scelta del tema: il tema è il layout grafico del sito e può essere scelto tra quelli disponibili sul repository ufficiale di WordPress o acquistati da fornitori terzi. In base alle esigenze del progetto, è possibile personalizzare il tema tramite il codice o attraverso l'utilizzo di plugin.
2. Configurazione del sito: una volta scelto il tema, è possibile configurare il sito definendo le pagine statiche, il menu di navigazione, i widget e le impostazioni generali (lingua, permessi degli utenti, SEO, ecc.).

3. Creazione dei contenuti: con il sito configurato, è possibile creare i contenuti delle pagine e degli articoli. Ogni contenuto può essere personalizzato attraverso l'utilizzo di tag, categorie, immagini e video.
4. Utilizzo dei plugin: i plugin sono componenti aggiuntivi che permettono di estendere le funzionalità di WordPress. È possibile scegliere tra i plugin disponibili sul repository ufficiale o acquistare quelli di fornitori terzi. Alcuni esempi di plugin utili possono essere quelli per la gestione delle email, per la sicurezza del sito o per la creazione di form.
5. Testing e messa in produzione: una volta terminata la fase di sviluppo del sito, è importante testarlo per verificare che tutto funzioni correttamente. Successivamente, è possibile mettere il sito in produzione su un server web.

Algebra Razionale

L'algebra relazionale è un linguaggio formale per descrivere le operazioni che possono essere eseguite sui database relazionali. Le operazioni fondamentali dell'algebra relazionale includono la selezione, la proiezione, il prodotto cartesiano, il join naturale, l'unione, la differenza e l'intersezione.

- Selezione: l'operazione di selezione consente di estrarre una parte delle righe di una tabella, in base a una condizione specificata. La sintassi dell'operatore di selezione è:

  $\sigma_{condizione}(nome - tabella)$
- Proiezione: l'operazione di proiezione consente di estrarre solo alcune delle colonne di una tabella, in base a una specifica selezione. La sintassi dell'operatore di proiezione è:

  $\pi_{elenco-colonne}(nome - tabella)$
- Prodotto cartesiano: l'operazione di prodotto cartesiano consente di combinare tutte le righe di due tabelle in un'unica tabella, con ogni riga della prima tabella abbinata a tutte le righe della seconda tabella. La sintassi dell'operatore di prodotto cartesiano è:

  $R \times S$
- Join naturale: l'operazione di join naturale consente di combinare le righe di due tabelle in base alle colonne che hanno lo stesso nome e lo stesso tipo di dati. La sintassi dell'operatore di join naturale è:

  $R \bowtie S$
- Unione: l'operazione di unione consente di combinare due tabelle in un'unica tabella, mantenendo solo le righe univoche. La sintassi dell'operatore di unione è:

  $R \cup S$
- Differenza: l'operazione di differenza consente di estrarre le righe di una tabella che non compaiono in un'altra tabella. La sintassi dell'operatore di differenza è:

  $R - S$
- Intersezione: l'operazione di intersezione consente di estrarre solo le righe che compaiono sia nella prima che nella seconda tabella. La sintassi dell'operatore di intersezione è:

  $R \cap S$

Accesso ai database (MySQL e SQL Server) tramite C#

In C#, è possibile accedere ai database MySQL e SQL Server tramite ADO.NET.

ADO.NET è una libreria di accesso ai dati che fornisce un'interfaccia unificata per accedere ai diversi tipi di database. Esistono due modalità principali di accesso ai dati tramite ADO.NET: modalità connessa e modalità disconnessa.

- Nella modalità connessa, l'applicazione mantiene una connessione aperta con il database durante l'interazione con i dati. La modalità connessa è particolarmente utile quando si lavora con un numero limitato di record o quando si effettuano aggiornamenti frequenti. In ADO.NET, la modalità connessa viene utilizzata principalmente tramite l'oggetto `SqlConnection`, che rappresenta una connessione a un database SQL Server o MySQL, e l'oggetto `SqlCommand`, che rappresenta un'istruzione SQL da eseguire sul database.

- Nella modalità disconnessa, i dati vengono recuperati dal database e memorizzati in un oggetto `DataSet`. L'applicazione può quindi lavorare con i dati memorizzati in memoria senza mantenere una connessione aperta con il database. La modalità disconnessa è particolarmente utile quando si lavora con un grande numero di record o quando si eseguono operazioni di sola lettura sui dati. In ADO.NET, la modalità disconnessa viene utilizzata principalmente tramite gli oggetti `SqlDataAdapter` e `MySqlDataAdapter`, che recuperano i dati dal database e li inseriscono in un oggetto `DataSet`.

- Esempio 1: Accesso ai dati in modalità connessa utilizzando ADO.NET in C#

```
using System.Data.SqlClient;

string connectionString = "Data Source=myServerAddress;Initial Catalog=myDataBase;User Id=myUsername;Password=myPassword;";

using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();

    string query = "SELECT * FROM Customers";
    SqlCommand command = new SqlCommand(query, connection);

    SqlDataReader reader = command.ExecuteReader();

    while (reader.Read())
    {
        string name = reader["Name"].ToString();
        int age = (int)reader["Age"];
        Console.WriteLine($"Name: {name}, Age: {age}");
    }

    reader.Close();
    connection.Close();
}
```

- Esempio 2: Accesso ai dati in modalità disconnessa utilizzando ADO.NET in C#

```
using System.Data.SqlClient;

string connectionString = "Data Source=myServerAddress;Initial Catalog=myDataBase;User Id=myUsername;Password=myPassword;";

string query = "SELECT * FROM Customers";
SqlDataAdapter adapter = new SqlDataAdapter(query, connectionString);

DataSet dataset = new DataSet();
adapter.Fill(dataset, "Customers");

DataTable customersTable = dataset.Tables["Customers"];

foreach (DataRow row in customersTable.Rows)
{
    string name = row["Name"].ToString();
    int age = (int)row["Age"];
    Console.WriteLine($"Name: {name}, Age: {age}");
}
```

NOSQL Database

I database NoSQL sono database non relazionali, utilizzati principalmente per gestire grandi volumi di dati in modo scalabile e distribuito. Tra i vari tipi di database NoSQL, uno dei più popolari è MongoDB, un database orientato ai documenti che utilizza BSON (Binary JSON) come formato di dati.

Ecco alcuni punti chiave per la conoscenza e l'utilizzo di MongoDB con C#:

1. Installazione e configurazione di MongoDB: prima di utilizzare MongoDB con C#, è necessario installarlo e configurarlo correttamente. MongoDB fornisce una documentazione dettagliata sul suo sito ufficiale, che può aiutare a installarlo e configurarlo in base alle proprie esigenze.

2. Utilizzo di MongoDB con C#: per utilizzare MongoDB con C#, è possibile utilizzare il driver ufficiale di MongoDB per .NET, chiamato "MongoDB.Driver". Questo driver può essere scaricato tramite NuGet, il gestore di pacchetti per Visual Studio.
3. Programmazione C# effettuando operazioni CRUD su MongoDB: una volta che il driver è stato installato, è possibile utilizzarlo per effettuare le operazioni CRUD (Create, Read, Update, Delete) su MongoDB. Ad esempio, per creare un nuovo documento, si può utilizzare il seguente codice:

```
var collection = database.GetCollection<BsonDocument>("collection_name");
var document = new BsonDocument
{
    { "key1", "value1" },
    { "key2", "value2" }
};
collection.InsertOne(document);
```

In questo esempio, "collection_name" è il nome della collezione in cui si vuole inserire il documento, mentre "key1" e "key2" sono le chiavi del documento e "value1" e "value2" sono i relativi valori.

Per la lettura dei documenti, si può utilizzare il metodo "Find":

```
var collection = database.GetCollection<BsonDocument>("collection_name");
var filter = Builders<BsonDocument>.Filter.Eq("key1", "value1");
var result = collection.Find(filter).ToList();
```

In questo esempio, "key1" e "value1" sono i criteri di ricerca per i documenti.

Per l'aggiornamento dei documenti, si può utilizzare il metodo "UpdateOne":

```
var collection = database.GetCollection<BsonDocument>("collection_name");
var filter = Builders<BsonDocument>.Filter.Eq("key1", "value1");
var update = Builders<BsonDocument>.Update.Set("key2", "new_value");
collection.UpdateOne(filter, update);
```

In questo esempio, si sta aggiornando il valore della chiave "key2" nel documento in cui la chiave "key1" ha il valore "value1".

Per l'eliminazione dei documenti, si può utilizzare il metodo "DeleteOne":

```
var collection = database.GetCollection<BsonDocument>("collection_name");
var filter = Builders<BsonDocument>.Filter.Eq("key1", "value1");
collection.DeleteOne(filter);
```

In questo esempio, si sta eliminando il documento in cui la chiave "key1" ha il valore "value1".

Progettazione software

- Esempio con spiegazione teorica di un semplice applicativo lato client che effettua chiamate ad un server che ha un web services REST. Spiegazione dettagliata dei compiti del client (controllo input con espressioni regolari), come passa i parametri al server, come si comporta il server, database e interazione, cosa restituisce, in che formato (JSON o XML) e come si comporta il client in base alla risposta ricevuta • Metodi GET, POST, PUT, DELETE loro comparazione.

La progettazione software è il processo di sviluppo di un'applicazione o di un sistema software, che prevede la definizione dei requisiti, l'analisi, la progettazione, l'implementazione, il testing e la manutenzione del software.

Ecco un esempio di un semplice applicativo lato client che effettua chiamate ad un server che ha un web services REST:

> Immaginiamo di voler sviluppare un'applicazione per la gestione di un elenco contatti. Il nostro sistema dovrebbe consentire all'utente di aggiungere, eliminare e modificare i contatti. Inoltre, dovrebbe essere possibile visualizzare l'elenco completo dei contatti o cercare un contatto specifico.
>
> Il nostro applicativo sarà composto da due parti: un client, ovvero l'interfaccia utente, e un server, che conterrà i dati e gestirà le richieste del client.

Il client sarà scritto in un linguaggio di programmazione lato client, come JavaScript, e utilizzando HTML e CSS per la presentazione grafica. Il client dovrà essere in grado di gestire l'input dell'utente, controllando che siano rispettati i criteri di validità tramite espressioni regolari, e di trasmettere i dati al server tramite chiamate API REST.

Le chiamate API REST sono richieste HTTP che permettono al client di interagire con il server e di scambiare informazioni con esso. Nel nostro esempio, il client dovrà utilizzare i metodi HTTP GET, POST, PUT e DELETE per gestire le richieste al server.

- Il metodo GET viene utilizzato per recuperare dati dal server, ad esempio l'elenco completo dei contatti o i dettagli di un singolo contatto.
- Il metodo POST viene utilizzato per inviare dati al server, ad esempio quando l'utente aggiunge un nuovo contatto.
- Il metodo PUT viene utilizzato per aggiornare i dati sul server, ad esempio quando l'utente modifica i dettagli di un contatto esistente.
- Il metodo DELETE viene utilizzato per eliminare i dati dal server, ad esempio quando l'utente cancella un contatto.

Il server sarà scritto in un linguaggio di programmazione lato server, come PHP o Python, e utilizzerà un database per gestire i dati dei contatti. Il server dovrà essere in grado di ricevere le richieste del client, elaborarle e rispondere con i dati richiesti. Inoltre, dovrà gestire eventuali errori e rispondere con un codice di stato HTTP appropriato.

Il server dovrà inoltre implementare un'API REST, ovvero un insieme di endpoint che il client potrà utilizzare per interagire con il sistema. Ad esempio, il server potrà fornire un endpoint per l'elenco completo dei contatti, un endpoint per recuperare i dettagli di un singolo contatto, un endpoint per aggiungere un nuovo contatto, un endpoint per modificare i dettagli di un contatto esistente e un endpoint per eliminare un contatto.

Il server dovrà anche decidere il formato in cui restituire i dati al client. Un formato comune è JSON (JavaScript Object Notation), un formato di dati leggero e facile da leggere che può essere facilmente interpretato da qualsiasi linguaggio di programmazione. Tuttavia, in alcuni casi, può essere necessario utilizzare anche altri formati, come XML.

Infine, il client dovrà essere in grado di elaborare la risposta ricevuta dal server e di presentare i dati all'utente in modo chiaro e intuitivo. In caso di errori, il client dovrà essere in grado di gestirli e di informare l'utente in modo appropriato.

In sintesi, la progettazione di un applicativo lato client che utilizza un web services REST prevede la definizione dei requisiti, l'analisi, la progettazione, l'implementazione e il testing sia del client che del server. È importante definire correttamente le chiamate API REST e i metodi HTTP da utilizzare, scegliere il formato dei dati e gestire eventuali errori.

Big Data

Il termine "Big Data" si riferisce a un insieme di dati estremamente voluminoso, complesso e diversificato che richiede tecnologie e metodi specifici per essere raccolto, gestito e analizzato. Questi dati possono provenire da fonti diverse, come social network, sensori, transazioni finanziarie, dati di navigazione web, dati meteorologici, dati medici e molti altri.

I Big Data sono caratterizzati da tre V: Volume, Velocity e Variety. Il Volume si riferisce alla quantità enorme di dati che devono essere gestiti, spesso in petabyte o exabyte. La Velocity si riferisce alla velocità a cui i dati vengono generati, raccolti e analizzati, che richiede una capacità di elaborazione in tempo reale. La Variety si riferisce alla diversità dei dati, che possono essere strutturati, non strutturati o semi-strutturati.

Per gestire i Big Data, sono necessarie tecnologie avanzate come Hadoop, Spark, NoSQL, Apache Cassandra e altre. Queste tecnologie consentono di distribuire i dati su cluster di server, elaborarli in parallelo e ottenere risultati rapidi e precisi.

L'analisi dei Big Data ha una grande importanza in diversi campi, come il marketing, la finanza, la medicina, la sicurezza e molti altri. L'analisi dei Big Data consente di identificare trend, anomalie, pattern e relazioni tra i dati, che possono essere utilizzati per prendere decisioni strategiche e per migliorare i processi aziendali.

Analisi dei dati con MS Excel pivot table e MS Power BI

Sia MS Excel pivot table che MS Power BI sono strumenti software utilizzati per analizzare grandi quantità di dati in modo efficiente e intuitivo.

- Le pivot table di Excel sono tabelle dinamiche che permettono di riepilogare e aggregare grandi quantità di dati in modo rapido e semplice. Con una pivot table, è possibile visualizzare i dati in diversi modi, come ad esempio totali, media, conteggi, percentuali e altro ancora, a seconda delle esigenze dell'analisi.

Per creare una pivot table in Excel, è necessario selezionare il set di dati di origine e definire le righe, le colonne e i valori della tabella pivot. Una volta creata la tabella pivot, è possibile filtrare e raggruppare i dati in modo da ottenere informazioni utili.

- MS Power BI, d'altra parte, è una piattaforma di business intelligence che permette di analizzare e visualizzare grandi quantità di dati da diverse fonti. Power BI consente di creare dashboard e report interattivi e personalizzati, utilizzando un'interfaccia intuitiva e facile da usare.

  Per creare un report in Power BI, è necessario connettersi alle diverse fonti di dati, come ad esempio file Excel, database, servizi cloud e altro ancora. Una volta connesse le fonti dati, è possibile creare diverse visualizzazioni, come grafici, tabelle, mappe e altro ancora. Infine, è possibile creare un dashboard che aggrega e visualizza le diverse visualizzazioni in modo intuitivo e interattivo.

In entrambi i casi, sia Excel pivot table che Power BI permettono di analizzare grandi quantità di dati in modo efficace, fornendo una panoramica completa dei dati e delle informazioni utili per prendere decisioni informate.